Structured Programming Language (CSE113)
Lecture #08
String in C programming

string: In C programming, array of characters is called a string. A string is terminated by a null character '\0'. Example: "Hello World"
Format Specifier: %s

char: A single symbol. Example: 'A' 'B' '1'
Format Specifier: %c

Declaration of strings: Strings are declared in a similar manner as arrays. Only difference is that, strings are of char type.
Example: char str[10];

Example: Write a C program to **read** and **write** your nickname

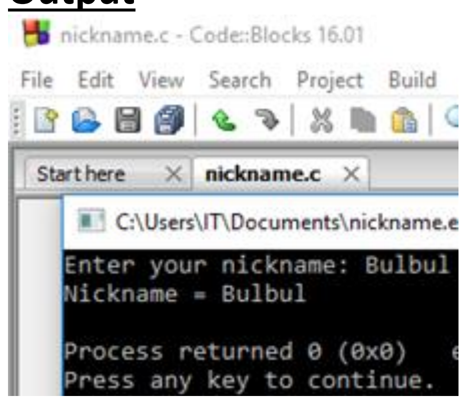
Source Code

```
#include<stdio.h>
int main()
{
   char str[100];

   printf("Enter your nickname: ");
   scanf("%s",&str);  //string input

   printf("Nickname = %s\n",str);  //string output
   return 0;
}
```

Output



Initialization of strings: In C, string can be initialized in a number of different ways.

- char str[50] = "BULBUL";
  '\0' would automatically be inserted at the end in this type of declaration
  OR
- char str[ ] = "BULBUL";
  OR
- char str[50] = {'B', 'U', 'L', 'B', 'U', 'L', '\0'};
  OR
- char str[] = {'B', 'U', 'L', 'B', 'U', 'L', '\0'};

\0 means null character

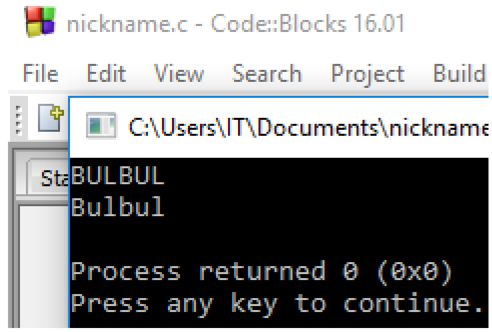
Example: Write a C program to **display** your nickname

Source Code

```
#include<stdio.h>
int main()
{
   char str[50] = {'B', 'U', 'L', 'B', 'U', 'L', '\0'};
   char str1[50] = "Bulbul";

   printf("%s\n",str);
   printf("%s\n",str1);
   return 0;
}
```

Output



gets() vs. puts()

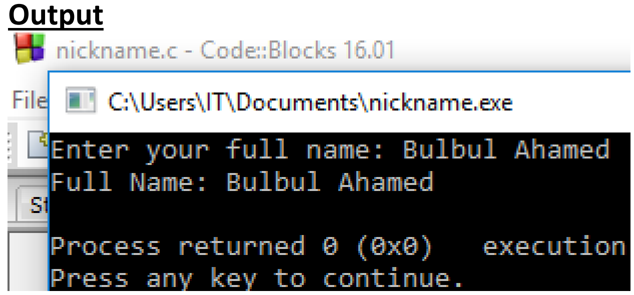
- **gets()** : Reads characters from the standard input and stores them as a string.
  Syntax for gets():
  `gets(char_array_variable);`
- **puts()** : prints characters from the standard output. Just like printf() function.
  Syntax for puts():
  `puts(char_array_variable);`

Example: Write a C program to **read** & **write** your full name using gets() and puts()

Source Code

```
#include<stdio.h>
int main()
{
   char str[100];
   printf("Enter your full name: ");
   gets(str);
   printf("Full Name: ");
   puts(str);
}
```

Output



String Length: **strlen()**

- **strlen()** function in C gives the length of the given string.
- **strlen()** function counts the number of characters in a given string and returns the integer value.

Example: Write a C program to find the **length** of a string

Source Code

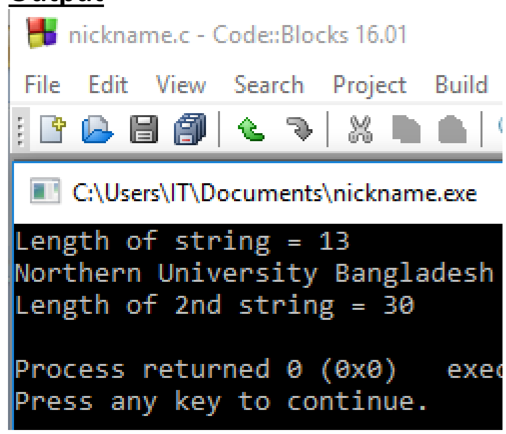
```
#include<stdio.h>
int main()
{
   char st[100] = "Bulbul Ahamed";
   char s2[50];

   printf("Length of string = %d\n",strlen(st));

   gets(s2);
   printf("Length of 2nd string = %d\n",strlen(s2));

   return 0;
}
```

Output



String Compare: **strcmp()**

- **strcmp()** function in C compares two given strings and returns zero if they are same. Negative value is returned when str1 < str2 and positive value is returned when str1 > str2.
- **strcmp()** function is case sensitive. i.e, "A" and "a" are treated as different characters.

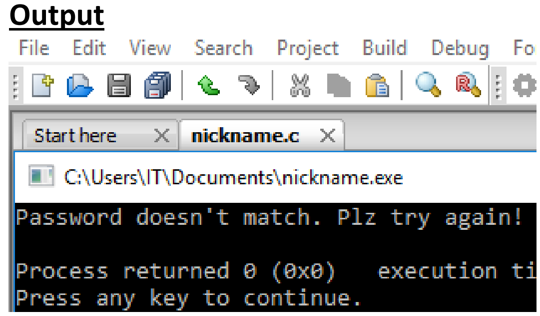
Example: Write a C program to compare two string whether they are same or not

Source Code

```
#include<stdio.h>
int main()
{
   char password[50] = "hello2018";
   char conpass[50] = "Hello2018";

   if(strcmp(password,conpass) == 0){
      printf("Login successful!\n");
   }else{
      printf("Password didn't match. Plz try again!\n");
   }
}
```

Output



Examples

1. Write a C program to print string characters one by one using Loop.

Source Code

```
#include<stdio.h>
int main()
{
   char str[100];
   int index;
   printf("Enter a string:\n");
   gets(str);
   for(index = 0; str[index] != '\0'; index++)
      printf("%c",str[index]);
   printf("\n");
}
```

Source Code

```
#include<stdio.h>
int main()
{
   char str[100];
   int index = 0;
   printf("Enter a string:\n");
   gets(str);
   while(str[index] != '\0'){
      printf("%c",str[index]);
      index++;
   }
}
```

Output

```
G:\C_Bkp\Desktop\C\string.exe
Enter a string:
I Love Bangladesh
I Love Bangladesh
```

2. Write a C program to print individual characters of string in reverse order.

Source Code

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str[100];
   int i;
   gets(str);
   int L = strlen(str) - 1;
   for(i = L; i  >= 0; i--)
      printf("%c", str[i]);
}
```

Source Code

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str[100];
   int i;
   gets(str);
   printf("%s", strrev(str));
}
```

Source Code

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str[100];
   int i;
   gets(str);
   int L = strlen(str);
   for(str[L] = '\0'; L > 0; L--)
      printf("%c", str[L-1]);
}
```

Output

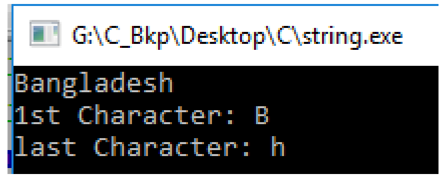
```
G:\C_Bkp\Desktop\C\string.exe
I Love Bangladesh
hsedalgnaB evoL I
Process returned 0 (0x0)
```

3. Write a C program to print 1st and last character of a string.

Source Code

```
#include<stdio.h>
#include<string.h>
int main(){
   char str[100];
   int i;
   gets(str);
   int L = strlen(str);
   printf("1st Character: %c\n",str[0]);
   printf("last Character: %c\n",str[L-1]);
   return 0;
}
```

Output



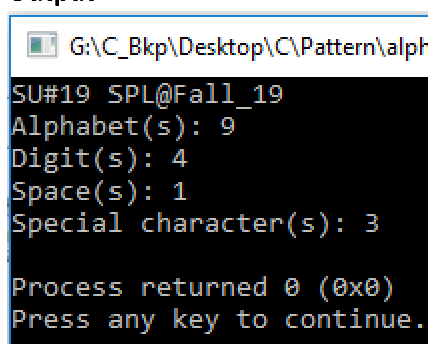
4. Write a C program to count alphabets, digits, spaces & special characters in a string.

Uses of **isalpha(), isdigit() & isspace()**

Source Code

```
#include<stdio.h>
int main(){
   char str[100];
   int alpha = 0, digit = 0, space=0, special = 0;
   gets(str);
   int i = 0;
   while(str[i] != '\0'){
      if(isalpha(str[i]) != 0 ){
         alpha++;
      }
      else if(isdigit(str[i]) != 0){
         digit++;
      }
      else if(isspace(str[i]) != 0){
         space++;
      }
      else{
         special++;
      }
      i++;
   }
   printf("Alphabet(s): %d\n", alpha);
   printf("Digit(s): %d\n", digit);
   printf("Space(s): %d\n", space);
   printf("Special character(s): %d\n", special);
   return 0;
}
```

Output



You can also write the following in while:

```
while(str[i] != '\0'){
      if(isalpha(str[i])){
         alpha++;
      }
      else if(isdigit(str[i])){
         digit++;
      }
      else if(isspace(str[i])){
         space++;
      }
      else{
         special++;
      }
      i++;
   }
```

5. Write a C program to count alphabets, digits, spaces & special characters in a string without using library functions.

Source Code

```c
#include<stdio.h>
int main()
{
    char str[100];
    int alpha = 0, digit = 0, space = 0, special = 0;
    gets(str);
    int i = 0;
    while(str[i] != '\0'){
        if((str[i]>='A' && str[i]<='Z')||(str[i]>='a' && str[i]<='z')){
            alpha++;
        }
        else if(str[i]>='0' && str[i]<='9'){
            digit++;
        }
        else if(str[i] == ' '){
            space++;
        }
        else{
            special++;
        }
        i++;
    }
    printf("Alphabet(s): %d\n", alpha);
    printf("Digit(s): %d\n", digit);
    printf("Space(s): %d\n", space);
    printf("Special character(s): %d\n", special);
    return 0;
}
```

Output

```
G:\C_Bkp\Desktop\C\Pattern\alph
SU#19 SPL@Fall_19
Alphabet(s): 9
Digit(s): 4
Space(s): 1
Special character(s): 3

Process returned 0 (0x0)
Press any key to continue.
```

isalpha() Return Value:

Return Value	**Remarks**
Zero (0)	If the parameter isn't an alphabet.
Non zero number	If the parameter is an alphabet.

isdigit() Return value:

Return Value	Remarks
Non-zero integer (x > 0)	Argument is a numeric character.
Zero (0)	Argument is not a numeric character.

isspace() Return value: This function returns a non-zero value(true) if c is a white-space character else, zero (false).

strlen() Return value: This function returns the length of string passed.

strcmp() Return value: This function return values that are as follows −

- if Return value = 0 then it indicates str1 is equal to str2.
- if Return value < 0 then it indicates str1 is less than str2.
- if Return value > 0 then it indicates str2 is less than str1.

Other necessary library functions:

- **strcat():** This function concatenates (joins) two strings.
- **strcpy():** The function copies the string to the another character array.
- **isalnum()**: This function checks whether the passed character is alphanumeric.
- **isupper()**: This function checks whether the passed character is lowercase letter.
- **islower()**: This function checks whether the passed character is an uppercase letter.
- **tolower():** This function converts uppercase letters to lowercase.
- **toupper():** This function converts lowercase letters to uppercase.
- **isxdigit():** This function checks whether character is hexadecimal.
- **isprint():** This function checks whether character is a printable character.
  There are many types of printable characters in C such as:
  - digits (0123456789)
  - uppercase letters (ABCDEFGHIJKLMNOPQRSTUVWXYZ)
  - lowercase letters (abcdefghijklmnopqrstuvwxyz)
  - punctuation characters (!”#$%&'()*+,-./:;?@[\]^_`{ | }~)
  - space ( )
- **ispunct():** This function checks whether character is a punctuation.
  All punctuation characters in C programming are:
  ! " # $ % & ' ( ) * +, - . / : ; ? @ [\ ] ^ _ ` { | } ~

Thank You ☺